Dynamic Adaptive Advance Bandwidth Reservation in Media Production Networks

Maryam Barshan, Hendrik Moens and Bruno Volckaert

Department of Information Technology, Ghent University – iMinds
Technologiepark-Zwijnaarde 15, 9052 Gent, Belgium
Email: maryam.barshan@intec.ugent.be

***Abstract*—Media production networks deal with large and predictable video transfers and streaming sessions which need to be scheduled to optimally use the limited network capacity. As the time and locality of transfers are predictable, advance bandwidth reservation results in more effective use of the underlying network. To offer reliable reservations, the incorporation of fault-tolerance related features in bandwidth reservation strategies resulting in redundancy is a necessity, but this incurs additional costs and extra performance overheads as network capacity remains unused to offer this protection. In order to mitigate these side-effects, we present an efficient event-driven runtime adaptive approach that continuously monitors the network transfers, dynamically updating the transfer schedule. This dynamic approach re-uses leftover network capacity to improve network utilization. The resulting management system offers protection from failures using resilient advance reservation, while also improving the network utilization and request admittance ratio. Our evaluation shows that the proposed approach leads to significant improvements, up to 13.9% in percentage of admitted requests in stable network conditions, compared to resilient advance reservation algorithms.**

***Index Terms*—Advance bandwidth reservation, media production network, runtime adaptation, network utilization.**

I. Introduction

The process of media production consists of multiple interactions and collaborations among different actors such as producers, directors, broadcasters, etc. and results in the transmission of large files such as raw video and audio files. The traditional way of distributing media content, such as physical transportation systems or dedicated point-to-point optical links, are highly inefficient in terms of installation time and cost. By using point to point dedicated links, resources are not shared and often underutilized. As such, deploying shared substrate networks that connect all collaborated end-sites over a large geographical area improves network utilization of media production networks.

In such a shared network, bandwidth is a valuable resource. Particularly for multimedia transfers, efficient bandwidth management is crucial [1]. In bandwidth-limited networks, a bandwidth scheduling mechanism needs to be designed to fulfill the QoS requirements, e.g. meeting deadlines and reliability. Bandwidth scheduling refers to bandwidth allocations with flexible options with regards to time and bandwidth requirements in both on-demand and in-advance reservation disciplines. Bandwidth reservation systems are generally capable of both advance and immediate reservations. The former allocate resources ahead of time in future time slots, while the latter reserve resources upon availability in the next immediate time slots. In media production networks, bandwidth requirements, timing constraint and locality of network transfers are mostly known hours or even days in advance. Consequently, deploying advance bandwidth reservation techniques leads to an increase in requests' admittance ratio and network utilization.

978-1-4673-9486-4/16/$31.00

In our previous work, we have presented deadline-aware optimal advance bandwidth reservation formulations [2] and efficient near-optimal equivalent solutions [3]. Two types of video transfer requests have been taken into account: file-based requests (FB) and video-streaming requests (VS). In case of file-based transfers, the reserved resources for a request may vary over time, as long as the delivery deadline is satisfied. For each video stream request, a constant duration and a fixed amount of reserved bandwidth is associated from source to sink of the request. The implicit idea in both approaches was to deliver the video files in earlier time slots and free up network capacities for possible requests that may be submitted in the future. Multiple requests in media production networks may depend on each other, meaning that one request can only start when other requests have been finished. This interdependence is explicitly incorporated in our approaches. Reliability has also been another concern of our previous work. In [4], we extend our advance reservation algorithms to provide a robust and resilient scheduling. This was a proactive approach meaning that scheduling is made robust through a protection mechanism. We try to find disjoint backup paths for the scheduled requests in advance, before any failure occurs in the network. This enables a fast reaction in case of a failure.

Based on discussions with industrial partners, we have found that reservations made for video streams, are not completely utilized throughout the requested time. Video streams can be resumed / played back multiple times during the reserved period, which causes idle reservations between resumes and playbacks. In our proposed approach, these unused capacities can also be exploited to transfer additional data. This means that we use these reserved capacities as double-purpose but video streaming sessions are prioritized. In doing so, as long as these reserved capacities are idle, extra data can be transferred, and as soon as a video stream gets active, an

event will be raised to prioritize the advance reservation made for this streaming session over the extra data transfers.

The proposed approach consists of two sequential parts. First, the network status and the reservation are being continually monitored and periodically updated. Second, the backup and unused network capacities, e.g. unused video stream reservations, are re-utilized to transfer more data than the schedule made by the resilient advance reservation algorithms. This leads to better utilization of substrate resources.

The remainder of this paper is organized as follows. We describe the work related to advance reservations and resilience in Section II. The presented method to improve network utilization is discussed in Section III. Section IV describes the proposed algorithms. The experimental results showing the efficiency of our approach are provided in Section V and we conclude in Section VI.

II. Related work

In order to support large-scale time-bound data transfers, there have been significant investigations in research and education networks, such as ESnet [5], Internet2 [6], TeraPaths [7] and VNOD [8]. In [9], a deadline-aware and flexible bandwidth reservation algorithm is proposed. However, these works differ from our work as they instead focus on generic data transfers and give little attention to video transfers with various requirements, which mainly exist in media production industry, as well as interdependencies among different transfers. According to [10], in order to tackle failures, two strategies can be distinguished: pre-failure and post-failure strategies. Burchard et al. in [11] consider a pre-failure-based strategy for advance reservations in grid environments. The authors in [12] present a fault-tolerant job scheduling approach for grid environments using adaptive task replication, which is a post-failure approach. Since meeting strict deadlines and QoS requirements is of great importance in our approach, using protection mechanisms tends to be more reliable.

In this paper, we propose a dual approach making partial use of our previous works [3] and [4]. In [3], we developed a deadline-aware advance reservation scheduling algorithm, customized for media production networks in which bandwidth scheduling algorithms are based on extending the classical shortest path and maximum flow problems. In [4], an extension of these algorithms is proposed to provide fault-tolerance, creating a reliable and resilient advance reservation scheduler by provisioning backup reservations for each flow. As redundancy imposes costs and resource waste, the main motivation of this paper is to mitigate the side-effect of using redundant reservations by employing them for data transfers as long as they are not needed for redundancy purposes.

III. Runtime adaptation methodology

Advance reservation scheduling has been proven to be a viable solution in media production networks as it allows the network manager to have efficient bandwidth management. In addition, reliability of data transfers in media production is similarly important. Resilient strategies enable reliable transmission of accepted requests without any loss in QoS in case of failure. In the resilient advance reservation algorithm, the backup paths are disjoint from the primary ones. The provisioned protection method guarantees a single link failure recovery. The backups are determined to fulfill the maximum bandwidth allocated on the links of the primary paths. This means that to provide 100% backup, there is no need to allocate the exact amount of bandwidth as in the primary paths. For instance, if two disjoint paths of 100Mpbs and 200Mpbs are allocated to a flow (300Mbps in total), 200Mbps is sufficient to be fulfilled by the backup paths. This is called shared protection [4].

The key idea of using the runtime adaptation approach is to benefit from the reliability and robustness provided by the resilient advance reservation algorithm and at the same time enhance network utilization and request admittance ratio. The runtime adaptation approach is a two-stage procedure which follows two sequential phases in every time slot: 1) the periodic update and 2) the periodic adaptation. The periodic update is the first step which takes into account the real transmitted data instead of the scheduled one and updates the schedule based on recent information by re-invoking the resilient AR scheduling algorithm. In stable network conditions, the actual transfers can potentially be ahead of schedule, due to the exploitation of reserved but unused capacities.

Periodic adaptation is a complementary step to continually adapt network transfers, taking into account the current state of network and transfers and making use of idle network capacity. The periodic update is repeated before the end of every time slot and the periodic adaptation before the start of the next time interval. In addition, the periodic adaptation algorithms are also triggered when other events occur: e.g. whenever a failure occurs, a file transfer is started, a file transfer is finished. These circumstances will be handled using an event-driven approach.

The runtime adaptation methodology consists of seven components as follows:

- **Advance reservation component**: in charge of producing an advance schedule using the resilient AR scheduling algorithm. The AR algorithm is invoked under two circumstances. First, when new scenarios enter the reservation system leading to an update of the entire schedule for all admitted and unfinished requests. Second, when the schedule needs to be updated during the periodic update. In both cases the schedule is modified at the start of the next time slot.
- **Global state manager**: contains all information about scheduling, network and request reservations, connections, demands, deadlines, etc. The time when the current time slot is started or when it gets finished can be retrieved from the global state.
- **Monitoring system**: keeps track of monitored times, residual demand and allocated bandwidth for all requests.
- **Job manager**: contains the list of current advance-scheduled requests and current waiting-list requests. Advance-scheduled requests refer to the requests that

have already been scheduled by the AR algorithm to be transferred in the current time slot. The waiting-list requests are those requests that can be started in this time slot, but are postponed due to limited network capacity.

- **Connection manager**: decides what to do when a transfer is started or stopped. Whenever a connection for a file transfer is terminated, the links that were in use by this connection become free. In order to improve network utilization, this capacity can be used by other active requests if shared links were in use. To achieve this, after completion of a file transfer, an event will be raised.
- **Reservation manager**: collects all the information about the reservations of each request. Primary allocations, backup reservations, extra allocations made during the periodic adaptation phase and allocated network resources can be retrieved from this component.
- **Adaptive optimization component**: in charge of optimization to try and push more data than what has been guaranteed through reservation. The Adaptive Optimization (AO) algorithm is the main algorithm in this component. Based on this algorithm, the current schedule is analyzed and adapted to use idle bandwidth capacities.

During the periodic update, first the current status of the network and transfers is monitored and then the resilient AR algorithm is invoked. This process updates the entire schedule based on the information retrieved from the monitoring system. This information is stored in the global state manager. Then the reservations are derived from the AR schedule and are set as advance-scheduled requests in the job manager. The list of advance-scheduled requests contains all requests which have been scheduled and are ready to be transferred. However, there are other flows which can potentially be started right now, but because of bandwidth constraints have been scheduled to be transferred in the future. These requests are kept in a waiting list to be processed during the periodic adaptation phase.

In periodic adaptation, the transfers are being continuously monitored to ensure that at least the advance-scheduled flows are completely transferred. Any request start / stop time or any failure will raise a specific event, which according to the event type is handled differently. For example, if a video stream is resumed, the extra data transfers that are currently being sent over that video stream reservation have to be interrupted. Also, any file-based transfer start / stop or failure will trigger the AO algorithm to rearrange the reservations. As such, during the periodic adaptation, the AO algorithm is triggered multiple times as long as there are active requests in the system.

IV. Runtime Adaptation algorithms

In this section the algorithms which are used in the periodic update and periodic adaptation phases of the runtime adaptation approach are described.

A. Periodic update algorithms

The periodic update phase consists of two algorithms: the *UpdateRequestsInfo* algorithm and the resilient AR scheduling algorithm. We do not elaborate on the resilient AR scheduling algorithm as it has already been explained in detail in [4]. In the *UpdateRequestsInfo* algorithm, the demand of submitted flows is updated. To achieve this, first finished and unadmitted requests are removed from the reservation system and the demand of all other submitted requests is updated based on the type of request. For file-based requests we deal with volume, so the allocated bandwidth is not fixed and may vary from one timeslot to another. In contrast, for video-steaming requests we deal with fixed and constant bandwidth requirements. Therefore, in this algorithm for file transfers, the last monitoring time, last allocated bandwidth and residual volume are updated based on the monitoring information. For video streams, the requests of which the deadlines have expired will be added to the list of removed requests. As our approach supports interdependencies among requests, all requests with active dependencies are also re-evaluated to check if they rely on removed requests.

B. Periodic adaptation algorithms

The Adaptive Optimization (AO) algorithm, which is frequently triggered in the periodic adaptation phase of runtime adaptation approach, is shown in Algorithm 1. This algorithm also triggers the *UpdateRequestsInfo* algorithm. As such, the demand of all requests is already updated whenever the AO algorithm is called.

Based on this algorithm, the advance-schedule requests and the list of waiting requests are retrieved from the job manager and the reservations for backups and video streams are ignored. Every video stream resume / play-back will trigger an event which leads to prioritizing the video streams over extra transfers. This provides us with a network in which only the primary reservations occupy the network capacities. Then sequentially for the requests in the advance-schedule requests list and waiting-list requests the following steps are performed: a new schedule for transfers over this residual graph is computed. Therefore, extra reservations will be made on top of the primary reservations and the assigned request bandwidth will thus potentially be increased. For each request, these new allocations will be updated in the reservation manager. Based on these new allocations, the start time and finish time of the requests are set and kept in the connection manager. All the requests' reservation and connection information is also saved in the global state manager.

The earliest finished request will raise an event. This event first cancels the stop time of all other active requests for which the finished times are set. Then, the AO algorithm is triggered to calculate new extra allocations and finish times. Since one request is already finished and the allocated resources are freed up, these new finish times tend to be earlier than the previously (now canceled) ones. This cycle is repeated as long as active requests trigger events. Detecting a failure / repair may also cause another type of event. The failed / restored network elements are removed from / restored to the network topology and the AO algorithm is re-invoked.

```
UpdateRequestInfo();
ASReq← JobManager.getASReq(current time);
WLReq← JobManager.getWLReq(current time);
PriorityBasedSorting (ASReq);
PriorityBasedSorting (WLReq);
NewGraph ← graph.remove(VSs, Backups);
ExtraAlloc ← BWallocation(ASReq, NewGraph);
for (rq ∈ ASReq) do
    TotalAlloc(rq)← PrimaryAlloc(rq) + ExtraAlloc(rq);
    Stop(rq) ← EstimateStopTime(TotalAlloc(rq),
    residualVol(rq));
    rq.SetReservations();
    rq.SetConnection("start", current time);
    rq.SetConnection("stop", Stop(rq));
end
ExtraAlloc ← BWallocation(WLReq, NewGraph);
for (rq ∈ WLReq) do
    if (ExtraAlloc(rq) ! = 0) then
        Stop(rq)←EstimateStopTime(ExtraAlloc(rq),
        residualVol(rq));
        rq.SetReservations();
        rq.SetConnection("start", current time);
        rq.SetConnection("stop", Stop(rq));
    end
end
```

Algorithm 1: Adaptive Optimization (AO) algorithm

V. Experimental results

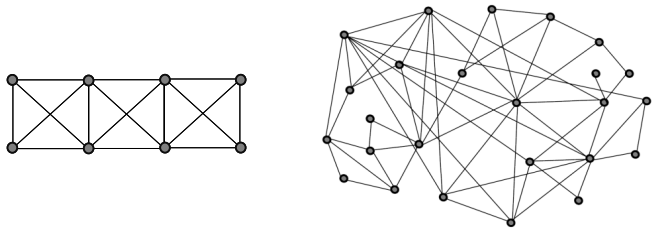
In order to model the dynamic aspect of our model, we have designed a discrete-event-based simulator using the Java-based MASON multi-agent simulation toolkit [13].

A. Evaluation Setup

In this evaluation we have used two topologies for media production networks which are depicted in Figure 1. We have defined three use case scenarios based on the information gathered from several Belgian media production actors. Each scenario contains a collection of interdependent file and video streaming requests. Use case 1 is composed of 5 different file transfer requests. Use case 2 comprises 18 interdependent file transfers. The third use case includes 4 file transfer requests and 4 video streams. The interactions between different media production actors in the use cases can be observed from [3]. A fixed time interval granularity of 1 hour is used and each simulation run covers a 24 hour period. All results are averaged over 50 runs with different randomized inputs, error bars denote the standard error. In all evaluations throughout this section, we assume that no failure has occurred and that video streams have been reserved but do not get activated. Throughout this section, *DARA*[*XX%*]+*RA* denotes that the resilient version of dynamic advance reservation approach (DARA) with *XX%* of backup is used. The second part (*RA*), is optional and refers to usage or non-usage of the runtime adaptation approach.

B. Impact of available bandwidth

For this evaluation, in the 8-node topology, the number of use cases equals 20, of which 7, 7 and 6 belong to use case 1, use case 2 and use case 3 respectively (209 requests in total) and for the larger network, the number of scenarios is 50, of which 17, 17 and 16 belong to the first, second and third use cases respectively (519 requests in total).



(a) 8-node topology (b) 25-node topology

Fig. 1: Media production networks used for evaluation.

Figure 2a and Figure 3a show the impact of various network capacities in 8-node and 25-node networks respectively. As can be seen in both figures, the runtime adaptation approach has noticeably improved request admittance ratio. These figures also show that having a higher resiliency percentage reduces the acceptance rate. Deploying the runtime adaptation approach, the percentage of admitted requests is improved up to 10.4% and 13.9% in Figure 2a and Figure 3a respectively.

C. Impact of network load

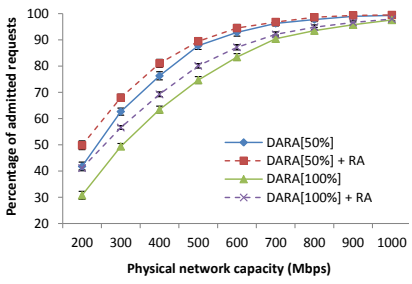
Figure 2b and Figure 3b shows the impact of network load on the performance of the runtime adaptation approach using the 8-node and the 25-node topology respectively. In both figures the network capacity of $300Mbps$ is used. Since the network capacities remain fixed in all experiments, adding more requests leads to an increase in rate of request rejection. The results show that for both smaller and larger topologies, the runtime adaptation approach improves the percentage of admitted requests up to 4.26% and 9.1% on average in Figure 2b and Figure 3b respectively.

D. Evaluation of execution times

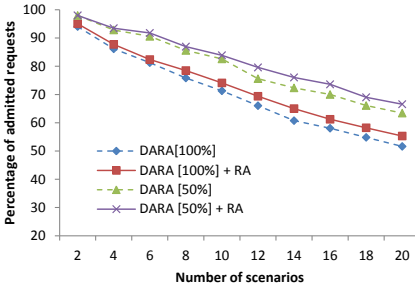
Figure 2c and Figure 3c compare the computational execution time of the resilient AR scheduling algorithm and the proposed runtime adaptation approach. As can be seen in both figures the percentage of resiliency has a low impact on execution time. Therefore 50% or 100% backup are within the same range with a negligible difference. Our results in Figure 2c indicate that deploying the runtime adaptation approach increases the execution time by 2.2 times when 20 scenarios are submitted to the reservation system to 10.2 times when only 2 scenarios are active. The same trend can be observed from Figure 3c. This figure shows that the resilient advance reservation algorithms without deploying the runtime adaptation approach executes between 2.3 to 8.4 times faster.

VI. Conclusion

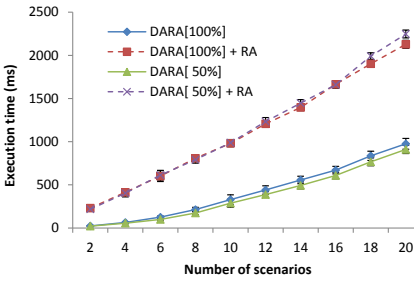
In our previous work, we have proposed a resilient advance reservation approach optimized for media production networks. This enables the reservation system to deliver reliable and consistent performance in the presence of failures. However, using redundancy imposes significant performance overheads and extra costs. In order to mitigate these side-issues,



(a) Impact of available bandwidth

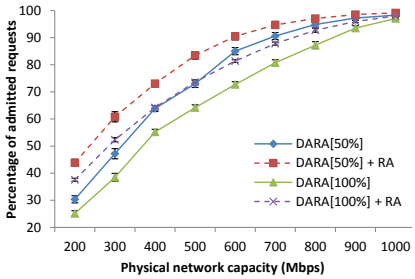


(b) Impact of network load

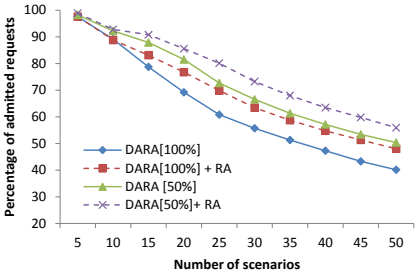


(c) Execution time

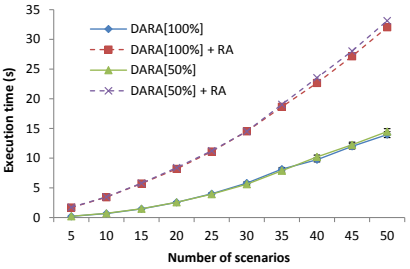
Fig. 2: Impact of the runtime adaptation approach on the performance of reservation system for 8-node topology



(a) Impact of available bandwidth



(b) Impact of network load



(c) Execution time

Fig. 3: Impact of the runtime adaptation approach on the performance of reservation system for 25-node topology

in this paper, we have designed and evaluated a dynamic event-driven approach aiming to increase network utilization and request admittance ratio. In this dynamic approach, a constant monitoring, adaptation and re-optimization is applied at runtime. We exploit underutilized network capacities to transfer more data than what has been scheduled as long as no failure is detected. Experimental results show that deploying this approach will noticeably increase the performance of the advance reservation system and the percentage of admitted requests up to 13.9% under stable network conditions.

ACKNOWLEDGMENT

The research leading to these results has been performed within the context of ICON MECaNO. This project is co-funded by iMinds, a digital research institute founded by the Flemish Government. Project partners are SDNSquare, Limecraft, VideoHouse, Alcatel-Lucent, and VRT, with project support from IWT under grant agreement no. 130646.

REFERENCES

[1] K. Nahrstedt and R. Steinmetz, "Resource management in networked multimedia systems," *Computer*, vol. 28, no. 5, pp. 52–63, 1995.

[2] M. Barshan, H. Moens, J. Famaey, and F. De Turck, "Algorithms for advance bandwidth reservation in media production networks," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 183–190, May 2015.

[3] M. Barshan, H. Moens, J. Famaey, and F. De Turck, "Deadline-aware advance reservation scheduling algorithms for media production networks," *Computer Communications*, 2015.

[4] S. Sahhaf, M. Barshan, W. Tavernier, H. Moens, D. Colle, and M. Pickavet, "Resilient algorithms for advance bandwidth reservation in media production networks," *DRCN2016*, 2015. Submitted.

[5] "Esnet: Energy sciences network." `http://www.es.net/`. Accessed: 2015-11-10.

[6] "Internet2." `http://www.internet2.edu/`. Accessed: 2015-11-10.

[7] B. Gibbard, D. Katramatos, and D. Yu, "Terapaths: end-to-end network path qos configuration using cross-domain reservation negotiation," in *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pp. 1–9, IEEE, 2006.

[8] D. Katramatos, S. Sharma, and D. Yu, "Virtual network on demand: Dedicating network resources to distributed scientific workflows," in *Proceedings of the Fifth International Workshop on Data-Intensive Distributed Computing Date*, DIDC '12, (New York, NY, USA), pp. 53–62, ACM, 2012.

[9] L. Shi, S. Sharma, D. Katramatos, and D. Yu, "Scheduling end-to-end flexible resource reservation requests for multiple end sites," in *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pp. 810–816, IEEE, 2015.

[10] L.-O. Burchard and M. Droste-Franke, "Fault tolerance in networks with an advance reservation service," in *Quality of ServiceâĂŤIWQoS 2003*, pp. 215–228, Springer, 2003.

[11] L.-O. Burchard, H.-U. Heiss, B. Linnert, J. Schneider, and C. A. De Rose, "Vrm: a failure-aware grid resource management system," *International journal of high performance computing and networking*, vol. 5, no. 4, pp. 215–226, 2008.

[12] B. Nazir, K. Qureshi, and P. Manuel, "Replication based fault tolerant job scheduling strategy for economy driven grid," *The Journal of Supercomputing*, vol. 62, no. 2, pp. 855–873, 2012.

[13] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.